信息检索关键技术及高性能检索系统设计

俞晓明 郭嘉丰 朱小飞 关峰 程学旗

摘要：网络等技术的快速发展，使人们能够访问的数据规模急剧增加。如何从海量信息中找到需要的信息成为难题。信息检索技术是应对该问题的有效手段，可以快速有效地帮助人们找到自己需要的信息。本文介绍了检索技术中使用的索引组织、检索模型、查询分析等关键技术及本课题组开发和维护的高性能开源检索系统 FirteX。

关键词：信息检索、检索模型、查询分析、排序学习（Learning to rank）、FirteX

1 引言

随着互联网信息数量的急速膨胀，信息检索作为一种有效的信息获取手段，在人们的日常生活中日益变得重要。广义的信息检索包括文本检索、图像检索、音视频检索等；狭义的信息检索是指文本检索或者文档检索，尤其指对非结构化（或半结构化）文本的检索，其任务就是研究如何从相对稳定的文本数据集中检索出与用户需求相关的文本。本文将主要针对文本检索相关的关键技术。

具体来说，信息检索完成的工作是根据用户的查询请求，在一个文档集中找出与用户请求最为接近的文档子集。右图给出了信息检索系统的一般处理过程。信息检索首先对文本建立索引。索引可以有效提高检索效率。检索时用户向检索系统提交查询，检索系统根据事先建立的索引进行检索，最后把检索到的文档根据一定的算法排序，按与查询请求相关度从高到低的顺序返回给用户。

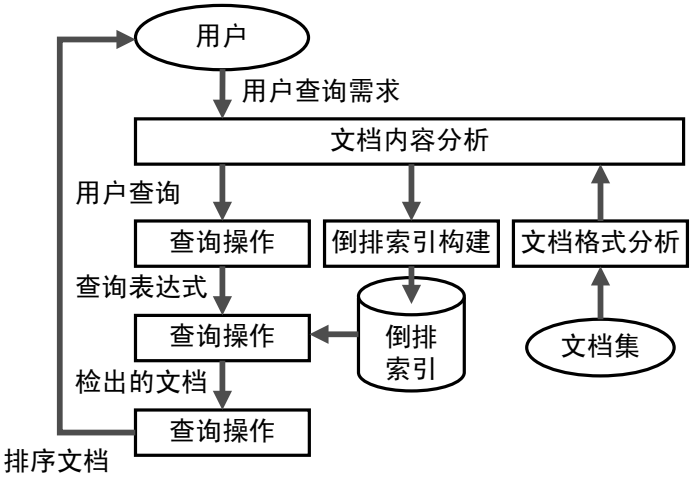

图1. 检索系统的一般处理过程

在信息检索中，查询请求是指用户对信息需求的描述，是用户信息需求的一种外在表现形式；文档是检索系统的基本检索对象或检索粒度，通常是由自然语言所描述的非结构化的自由文本或半结构化的文本，如网页、文字新闻、学术出版物、产品描述、博客页面等；文档集指的则是一定数目文档的集合。根据文档类型的不同，文档集会随时间发生频度不同的改变。例如，数字图书馆可能几天添加一些新书或者移除一些旧书，而论坛的帖子、博客的文章和上架的商品可能几分钟就会发生更新。根据文档集更新的快慢和方式，又可将文档集分为静态文档集、增量文档集和动态文档集。

在面向 Web 的信息检索系统中，除上图的一般处理过程，通常还会包含另外两个重要的模块：信息采集和信息抽取。它们都是为得到检索用的文档集做准备的过程。信息采集的

任务是从网上获取信息，通常使用网络爬虫完成。信息抽取完成的工作是从采集到的半结构化数据获取用于建立索引的结构化数据。

信息检索的研究范围十分广泛，涉及信息采集、表示、组织、存储、访问和搜索等[1]。近年来，随着云计算的流行，越来越多的研究者开始设计基于云计算平台的信息检索系统。这些系统可以很好地适应大规模数据的检索需求。但由于它们的基本原理与图 1 给出的一般处理过程相比没有发生大的变化，我们将不再深入到细节中。

本文将从索引组织、检索模型、查询分析等角度对信息检索系统的关键技术进行介绍。其中，第二节针对不同文档集分别介绍不同的索引组织方法；第三节给出了常用的检索模型，同时介绍了排序学习技术的进展；第四节总结了查询分析相关技术。最后，在第五节介绍本课题组开发的高性能开源检索软件 FirteX 的设计。

2 索引组织

第一节已提及，为文档集建立索引是为加快完成检索任务的速度[1]。针对不同需求，常见的索引组织方式有：签名档、位图文件和倒排索引等，其中倒排索引是信息检索中最常见的索引形式。

签名档（Signature File）是一种基于概率方法的文本索引结构[2]，它将每篇文档用一个签名来表示。签名由一系列的位掩码组成，在一定程度上描述了文档的内容。位图（Bitmap）是一种很简单的索引结构。对于词典中的每个词，都用一个位向量表示，其中每一位对应一篇文档。如果这个词出现在文档中则设为 1，否则设为 0。倒排索引是面向单词的索引机制。倒排索引由词汇表和事件表两部分组成。词汇表中的每个元素均称为词或索引项，事件表则是倒排链的集合，每个词对应一个倒排链。倒排链一般也称作后缀链，包含词所在文档的编号、出现频率和出现位置等信息。也就是说，文档的原始形式是文档包含词的关系，而倒排索引的构建就是将这种关系颠倒过来，变成词包含文档的关系，这也是“倒排”名称的由来。图 2 给出了简化的倒排索引的例子。

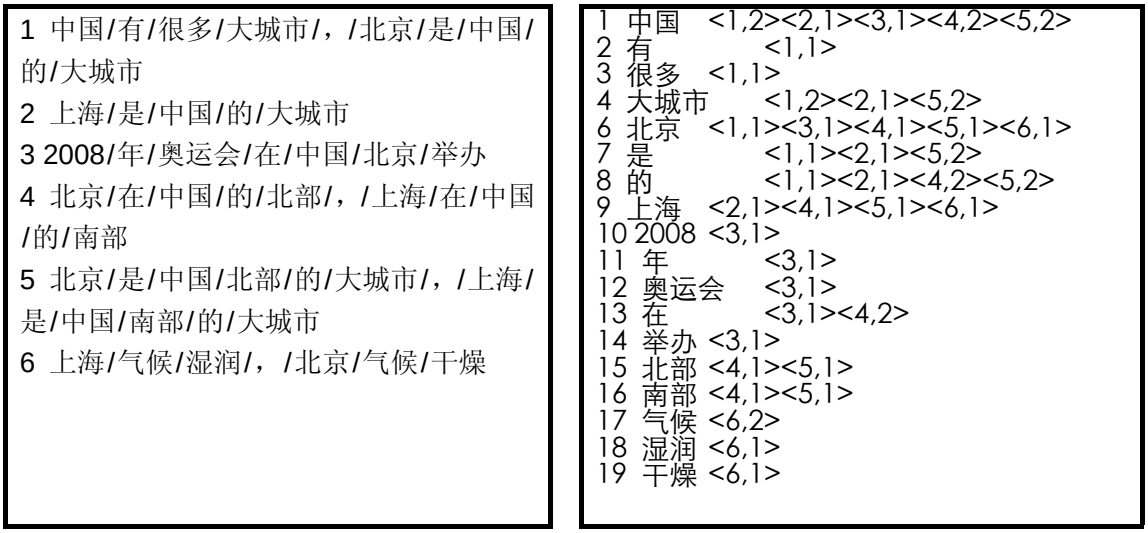

图2. 倒排索引的例子

倒排索引的构建需要大量的 CPU、内存和磁盘资源。同时，为了提高检索效率，减少

磁盘操作，一般将一个倒排链表存放在一个连续的磁盘空间。在实际情况下，文档集往往非常大，其索引远远超过了能完全放入内存的大小上限。因此，我们希望借用磁盘空间来构建大规模文档集。下面我们根据文档集的不同分别讨论倒排索引的建立方法。

2.1 静态文档集的索引

静态文档集中的文档不发生变化，索引建立过程可以一次或多次访问文档集来完成。总体来说，借助磁盘空间构建大规模文档集的倒排索引主要有三种方法：基于排序的倒排索引构建、两趟（多趟）倒排索引构建、一趟内存倒排索引构建。

维顿（I.H. Witten ）等人[2]认为倒排索引的构建实际上是一个排序的过程，原始的文档可以看成是由三元组$\langle t,d,f_{d,t}\rangle$组成的单元，其中$t$表示词，$d$表示文档编号，$f_{t,d}$表示该词在文档$d$中出现的频率。如果考虑词的位置，则每个三元组还对应一个词位置序列$\langle w_1,w_2,\cdots,w_{f_{d,t}}\rangle$。整个文档集可以看成是多个有序三元组$\langle t,d,f_{d,t}\rangle$的集合，因此倒排索引的构建只需要将上述三元组按照$t$排序，$t$相同再按$d$排序即可。

两趟（多趟）倒排索引构建方法最初由福克斯（E. Fox）和李（W. Lee）提出[3]，并由维顿等人[2]改进。该方法通过多趟扫描文档集构建倒排索引。第一趟扫描生成整个文档集的词典，并统计文档集中各个词的详细信息。经过一趟完整的文档集扫描，除了在内存中生成了整个文档集的完整词典外，也准确地知道了各个词的倒排链长度，可以预先分配各个词的倒排链的磁盘存储空间。在后续的文档集扫描中，直接将词的倒排链填充到预先分配的空间中。这种方法能保证单个词的倒排链在磁盘上连续存放。

在一趟内存倒排索引构建方法中，对每个词维护一个链表或动态增长的数组，用于记录该词的所有三元组$\langle t,d,f_{d,t}\rangle$。在处理过程中，各个词的倒排链是持续增长的。当倒排链的增长导致内存耗尽时，则将内存中的所有词及其倒排链写入磁盘，形成临时子索引（也称为run），释放内存空间，准备构建下一批文档的倒排索引。上述过程循环执行直到所有文档处理完毕。这时磁盘上存在多个临时子索引，再采用多路归并将所有子索引合并，形成最终的倒排索引。该方法由海因兹（S. Heinz）和 佐贝尔（J. Zobel）等人首次提出[4]。文献[2]、[4]、[5]给出了以上索引构建方法的对比，指出最高效的是一趟内存倒排索引构建方法，其规模可扩展性也非常好，可以处理任意大小的数据集。

2.2 动态文档集的索引

动态文档集会随着时间变化发生增删和修改。在文档集发生变化时，为了保证检索结果的正确性，索引也要进行相应的更新。由于索引的更新一般是在提供检索服务的同时进行，所以索引更新策略必须同时兼顾索引的效率和检索的效率。更新倒排索引最直接的方法是丢弃旧的倒排索引，重新扫描文档集以建立新的索引。这种方法称为索引重建。除此之外，常见的索引更新方法还有原地索引更新、基于合并的索引更新方法等。

原地索引更新方法的基本思想是：当文档集内的文档发生变化时，单独更新倒排索引中发生变化的倒排链，而其他没有发生变化的倒排链则保持不变。在实际应用中，文档的索引数据首先累积在内存中。当内存不够时，将内存中所有词的倒排链更新到磁盘上相应的倒排链中。倒排链的原地更新需要考虑倒排链的空间管理问题。将倒排链累积在内存中可以大幅度降低磁盘上倒排链的更新次数。在较少文档更新情况下，原地索引更新方法具有不错的性能。随着数据规模的增长、更新频率的加快，原地索引更新的性能会急剧下降[6]。

基于合并的索引更新方法首先在内存中建立文档的倒排索引。当内存不够时，依次读取

磁盘上每个词的倒排链，与内存中对应词的倒排链合并，再写入新的磁盘位置。所有词的倒排链合并完成后，在磁盘上形成新的倒排索引。在基于合并的索引更新方法中，需要读取磁盘上所有的倒排链，不管这些倒排链有没有进行更新。最后生成的新索引的所有倒排链也是按照词的递增顺序连续存储。用户查询处理过程中，单个词的倒排链只需要一次磁盘定位和读取操作，可以最大限度提高查询处理的性能。基于合并的索引更新，根据合并方式、磁盘上最多可允许的子索引数目等因素的不同可以分为多种方法：只允许磁盘上存在一个子索引的更新方法有立即合并更新方法；允许磁盘上存在多个子索引的更新方法主要有无合并策略；此外还有介于立即合并和无合并策略之间的对数合并策略、几何分割合并策略等。

（1） 立即合并

立即合并索引更新方法也称为再合并索引更新方法。顾名思义，就是当内存子索引需要迁移到磁盘上时，如果磁盘上已存在倒排索引，则将内存子索引与磁盘子索引立刻进行合并，生成新的索引替代旧索引。这种索引更新方式在磁盘上始终只保持一个子索引，每个词的倒排链都是顺序、连续存储，因此可以最大限度保障查询处理性能。但是，每次索引合并操作都需要读取、处理整个磁盘子索引，不管其中的倒排链是否已更新。因此，索引构建的代价较大。从表面看来，因为在每次的索引合并中都需要重新读取、处理已有的磁盘子索引，索引更新的性能可能很差。莱斯特（N. Lester ）等人[6] [7] 对立即合并索引更新方法和原地索引更新方法进行了实验对比和分析。他们的结论是，在很多实际场合中，立即合并索引更新方法的性能反而要比原地索引更新方法的性能更优。

（2） 无合并索引更新方法

在无合并索引更新方法中，当内存耗尽时将内存倒排索引直接写入磁盘，形成新的磁盘子索引，不作任何索引合并的操作，在磁盘上保持多个子索引。在检索过程中，词的倒排链的获取通过读取所有磁盘子索引中对应的倒排链完成。该方法对磁盘上的每个子索引只需要做一次写操作，可以使索引更新的性能最佳。但是，每个词的倒排链可能分布在多个磁盘子索引中，进行查询处理时需要多次磁盘定位和读取操作，影响查询处理的性能。

（3） 对数索引更新方法

立即合并和无合并的索引更新方法是在线索引更新的两种极端方式，只能单方面提高查询处理的性能或索引更新的性能。动态文本在线索引构建往往需要同时考虑查询处理和索引更新的性能，要求既具有较好的索引更新性能，又保证查询处理的性能不会下降太多。布舍尔（S. Büttcher）提出了一种对数合并索引更新方法[8]，引入了索引“代”的概念。从内存中迁移到磁盘上的子索引称为第 0 代子索引，由第 0 代子索引合并生成的子索引称为第 1 代，依此类推。同一代中不允许同时出现两个子索引，如果出现则触发一次合并事件，生成新一代子索引。如果新生成的子索引导致同一代又出现两个子索引，将再触发一次合并事件，这两次合并操作合并成一次进行。文献[8]分析了对数索引更新方法的索引更新和查询处理代价。

（4） 几何分割索引更新方法

莱斯特提出了一种类似于对数合并索引更新的策略[9]，称为几何分割索引更新方法。其基本思想是将待索引的数据集分为多个数目可控的分块 (Partition)，每个分块的大小遵循一定的几何规律。详细描述如下：假设内存中可容纳 $\boldsymbol{b}$ 篇文档，引入参数 $\boldsymbol{r}$ ，每个分块遵循下面的规则：每级分块包含的文档数目至多为 $(\boldsymbol{r}-1)\boldsymbol{b},(\boldsymbol{r}-1)\boldsymbol{rb},(\boldsymbol{r}-1)\boldsymbol{r}^2\boldsymbol{b},\cdots,(\boldsymbol{r}-1)\boldsymbol{r}^{k-1}\boldsymbol{b}$ 。

其中 $\boldsymbol{k}$ 表示第 $\boldsymbol{k}$ 级分块。在第 $\boldsymbol{k}$ 级，分块要么为空，要么至少包含 $\boldsymbol{r}^{k-1}\boldsymbol{b}$ 篇文档。调整

参数 r 可以一定程度地调节索引更新和查询处理的性能平衡。和对数合并索引更新方法一样，几何分割索引更新方法既不会有太频繁的索引合并操作，磁盘上也不会有太多的子索引，既可以保证索引更新的性能，查询处理的性能也不会下降太多。关于几何分割索引更新方法的详细分析和讨论，可以参考文献[9]。

（5） 基于动态平衡树的索引更新方法

郭瑞杰[10、33]提出了基于动态平衡树的索引更新策略。动态平衡树是一棵 m 叉树，且第 k+1 层节点的大小大约是第 k 层节点大小的 c 倍(m 和 c 是算法参数)；如果一棵树中所有结点都满足该条件，称这棵索引合并树是平衡的。在基于动态平衡树的索引更新算法中，新加入的子索引根据大小加入树的对应层。如果新索引的加入导致树中某些结点不满足平衡条件，则以通过合并子索引的方式恢复树的平衡。在该算法中，索引合并操作基本只在同一层的节点之间进行，可以保证参与合并的各个子索引大小相近，从而保证索引合并的效率。引文[10]的实验分析表明，基于动态平衡树的索引合并算法具有较以上索引合并算法更好的性能。

3 检索模型

在信息检索中，文档排序是核心。“排序”就是在一个文档集合中，对于给定的查询，使用检索模型给每篇文档打分，然后按照每篇文档的得分从高到低排序，排序的次序代表了文档与查询之间的相关性。信息检索中的排序方法起源于传统文本检索领域的排序方法，如向量空间模型(Vector Space Model)，概率模型（如 Okapi BM25）和语言模型(Language Model)。链接分析(如 Pagerank、HITS[12,13])的出现极大地提高了 Web 检索的有效性。近年来，排序学习(Learning to Rank)已经成为检索领域的一个热点问题。

3.1 传统排序模型

传统的排序模型中，比较具有代表性的方法有：向量空间模型，概率模型（Okapi BM25），语言模型以及链接分析模型(Pagerank、HITS)。

3.1.1 向量空间模型

向量空间模型是以词汇向量的形式表示文档，通过计算向量的相近程度来判断文档的相关性。在检索过程中，查询和文档都被转化成词的向量表示，通过计算向量之间的相似性(如夹角余弦)得到查询和每个文档的相似度。向量权重的计算通常使用 TF-IDF，其中 TF 是词在文档中出现的次数，IDF 是逆文档频率。该模型的优点是计算简单，缺点是使用的大都是经验公式，缺乏理论验证。

3.1.2 概率模型

在概率模型中，为了表示文档 D 和查询 Q 是否相关，设相关性 R 取值为 r 或 $\bar{r}$，分别表示 D 和 Q 相关或者不相关。概率检索模型的目标就是估计文档 D 和 Q 相关的概率，即 $P(R=r|D,Q)$。一般使用相关概率和不相关概率的比值来计算文档和查询的相关性，即

$$log\frac{P(r|D,Q)}{P(\bar{r}|D,Q)}$$

Okapi BM25 是一种著名的概率模型，Okapi BM25 的相似度计算公式为：

$$\sum_{w\in q\cap d}\left(ln\frac{N-df(w)+0.5}{df(w)+0.5}\times\frac{(k_1+1)\times c(w,d)}{k_1((1-b)+b\frac{|d|}{avdl})+c(w,d)}\times\frac{(k_3+1)\times c(w,d)}{k_3+c(w,d)}\right)$$

其中 d,q 分别表示查询和文档，$|d|$ 是文档 d 的长度，$avdl$ 是文档集合中文档的平均长度。w 表示特征词项，$c(w,d)$ 和 $c(w,q)$ 分别表示 w 出现在 d 和 q 中的个数，N 是文档集合中的文档总数，$df(w)$ 表示出现 w 的文档个数。$P(w|C)$ 表示 C 中出现 w 的概率。公式中的 k_1,k_3,b 都是人工经验调节的参数。概率模型的优点在于具有良好的数学理论基础，缺点在于无法处理语言中的长距离依赖关系。

3.1.3 语言模型

语言模型把相关度看成是每篇文档对应的语言下生成该查询的可能性，对于一篇文档 $d=w_1,w_2,\cdots,w_n$，统计语言模型是指概率 $P(w_1,w_2,\cdots,w_n)$。根据贝叶斯公式，有：

$$P(w_1,w_2,\cdots,w_n)=P(w_2,\cdots,w_n|w_1)=\cdots=P(w_1)\prod_{i=2}^{n}P(w_i|w_{i-1},\cdots,w_1)$$

通常采用一元模型，即假设文档中各个词都是独立的，此时

$$P(w_1,w_2,\cdots,w_n)=\prod_{i=1}^{n}P(w_i)$$

对于给定的查询，我们可以计算各个文档生成该查询的概率 $P(w|D)$，并以此作为判断文档与查询相似性的依据。由于存在数据稀疏性问题，计算结果可能出现零概率。解决的方法是使用数据平滑技术，为所有未观测到的特征词项分配一个大于零的概率。

常用的平滑方法包括线性插值平滑（亦称 Jelinek-Mercer 平滑）方法、狄利克雷（Dirichlet）平滑方法以及绝对折扣（Absolute Discount）平滑方法，该方法通过将已观测到的词项的频度减去一个常量的办法来降低已发生事件的概率。

3.1.4 链接分析模型（Pagerank、HITS）

链接分析模型通过网页间权值的传送，得到各个网页的重要性，重要性越大，排序越靠前。代表性的方法有 Pagerank 和 HITS 方法，两者之间的区别在于 PageRank 权值的传送是直接在权威（authority）型网页间进行的；而 HITS 在权值转送时，权威型网页的权值需经过目录（hub）网页的传递再进行传播。

3.2 排序学习

随着信息检索研究的不断发展，人们逐渐意识到，有太多的因素会影响到排序，仅仅依赖于文本和链接的排序方式具有很大的局限性。如果把这些相关的因素看成特征，并能够用某种机器学习的方式综合考虑，得出一个最合理的排序函数，将能够有效地提高排序的效果，这也就是排序学习要解决的最根本问题。目前，排序学习研究的趋势主要是从基于数据点（Pointwise）的排序方法到基于有序对（Pairwise）的排序方法，再到基于列表（Listwise）的排序方法，从查询无关到查询相关的进一步发展。

3.2.1 基于数据点的排序方法

基于数据点的排序方法是最早提出的，其基本思想是将查询/文档对（query/document）看成是一个训练样本，查询与文档的相关程度看成是该训练样本得分或类别。这样就可以将

排序问题转化成为一个回归问题或分类问题进行求解。代表性的方法有 PRank[14]和 MCRank[15]方法。

PRank (Perceptron Rank)方法将查询/文档对看成是训练样本，每个训练样本对应其相应的排序等级。PRank 的主要思想是：将训练样本映射到相应的实数值，同时每个排序等级对应于一个实数区间，这样就可方便地得到各个训练样本的排序等级。PRank 通过调整一个权重向量来使得排序损失最小化。理论上可以证明该方法是保序的、并且具有错误边界(mistake bound)，这两个性质确保了 PRank 算法的正确性和收敛性。

MCRank 方法将排序问题看成是一个多分类问题来处理，使用多个分类器的综合结果来确定最终的排序。在 MCRank 方法中，首先对训练样本（查询/文档对）进行分类，得到其所在类别，例如将训练样本归入到 K 个类别 $\{0,1,2,\cdots,K-1\}$ 中的一个类别，然后按照类别进行排序。对于类别相同的训练样本，其排序可以任意，这就造成使用 MCRank 时，排序结果不稳定。为了避免此问题，可以对每一个训练样本进行软分类（soft classification），得到训练样本的类别分布，然后根据其期望相关性对其进行打分，并将训练样本按照得分降序排列。

3.2.2 基于有序对的排序方法

基于有序对的排序方法利用文档之间的相关性的相对大小的关系来构造训练样本， 即每个样本由一对文档组成，若前一个文档比后一个文档更相关，其类别为正，否则类别为负。这样就可以对新来的一对文档进行分类，若结果为正，则表明前一个文档比后一个文档更相关；若结果为负，则表明后一个文档比前一个文档更相关。代表性的方法有：RankNet[16]，RankBoost [17]，Ranking SVM[18]。

伯格斯（C. Burges）和沙克德（T. Shaked）等人给出了一种简单的概率代价函数(cost function)，在此基础上提出了 RankNet 方法，在其研究中使用神经网络来对代价函数进行建模，然后使用梯度下降方法进行优化。设 $x \in \mathbb{R}$ 为样本的特征向量，函数 $f:\mathbb{R}^d \to \mathbb{R}$ 将样本映射到实数，即用实数来描述样本的排序， $f(x_i) > f(x_j)$ 表示 x_i 的排序比 x_j 更靠前，用 $x_i \rhd x_j$ 表示。定义 $P_{ij} \equiv P(x_i \rhd x_j) = exp(o_{ij}) / (1+exp(o_{ij}))$，其中 $o_{ij} \equiv (f(x_i) - f(x_j)) \in \mathbb{R}$。RankNet 使用交叉熵(cross entropy)作为损失函数，即

$$C_{ij} \equiv C(o_{ij}) = -\bar{P}_{ij} \log P_{ij} - (1-\bar{P}_{ij}) \log (1-P_{ij})$$

用来度量预测概率 P_{ij} 与目标概率 $\bar{P}_{ij}$ 之间的差异。

弗罗因德（Y. Freund）等人提出 RankBoost 方法，其基本思想源自于 AdaBoost 算法。RankBoost 是综合多个弱排序函数来得到最终的排序函数，排序函数为 $H(x) = \Sigma_{t=1}^{T} w_t h_t(x)$，其中 h_t 为第 t 个弱学习器， w_t 为相应弱学习器的权重。为了使学习器能够体现文档对的相对排序关系信息，即对于任意两篇文档 x_1, x_2，学习器既要体现文档 x_1 的排序是在文档 x_2 前面还是后面，同时还要体现出该排序的重要性或者说是强度。他们使用反馈函数 $\Phi: \chi \times \chi \to \mathbb{R}$ 来表示相对排序关系信息，当 $\Phi(x_1, x_2)>0$ 时，表示文档 x_1 的排序比文档 x_2 更靠前，当 $\Phi(x_1, x_2)<0$ 时，表示文档 x_1 的排序比文档 x_2 更靠后。若 $\Phi(x_1, x_2)=0$，则表示文档 x_1 和文档 x_2 的排序没有先后关系。 $|\Phi(x_1, x_2)|$ 的大小表示这种关系的重要性， $|\Phi(x_1, x_2)|$ 越大，表示文档 x_1 排序比文档 x_2 更靠前的关系越重要。在定义反馈函数 Φ 之后，问题就转化为要使错误文档对的数目（也可以是加权后的值）最小化。

Ranking SVM 的基本思想是将排序问题转化为二分类问题，然后使用支持向量机对问题进行求解。其过程主要分为两步：

第一步，类似于 RankNet, 在 Ranking SVM 中，首先使用排序函数 $f(x)$，（一般选择线性函数，如 $f(x)=\langle w \bullet x \rangle$，其中 w 为权重向量），将样本映射到实数值，定义 x_i 与 x_j 之间的序关系：

$$x_i \succ x_j \equiv f(x_i) - f(x_j) = \langle w \bullet (xi - xj) \rangle > 0$$

第二步，使用支持向量机模型来求解该二分类问题，对应的二次优化问题表示为：

$$min_w \Sigma_{k=1}^{l} \left[1 - Z_k \left\langle w \bullet (x_i^k - x_j^k) \right\rangle \right] + \lambda \|w\|^2$$

其中，第 k 个文档对由第 i 个和第 j 个文档组成，l 是文档对的数目，Z_k 表示第 k 个文档对的真实类别。

3.2.3 基于列表的排序方法

基于列表的排序方法是以每个列表为训练样本，一个列表本身就包含了一些排好序的文档，某些关系已经隐含在列表里面。基于列表的排序根据建立目标函数的角度不同一般可以分为两类：一种是使损失函数（loss function）最小化方法。这一类中的不同方法之间的主要区别在于其使用的损失函数不同。如 RankCosine 使用得分列表之间的余弦相似度来作为损失函数，ListNet 的损失函数是得分列表之间的库尔贝克-莱布勒（Kullback-Leibler）距离，ListMLE 则使用负对数似然值来作为损失函数；另一种是直接优化信息检索的度量指标(MAP、NDCG）方法，例如 AdaRank、SVM-MAP、SoftRank、LambdaRank。

在评价排序结果好坏的时候，基于列表的排序方法会把查询词对应的所有文档都考虑进去，而且可以将文档之间的关系，如相似度等建模，因此可以定义更加有效的排序函数；另外，由于是面向整个列表，它可以充分利用文档在列表中的位置信息，从而更加突出排在前面的文档。

4 查询分析

“查询”作为用户表达其信息需求的主要手段，是用户与信息检索系统之间沟通的桥梁。理想情况下，信息检索系统能够正确地理解用户的查询，在此基础之上采取合适的检索方式，返回相应的结果来满足用户的信息需求。然而，就如同计算机对自然语言的理解存在困难一样，检索系统对于作为用户与信息检索系统交互语言的查询的理解也同样面临巨大的挑战。近年来，随着信息检索技术的不断发展，对用户查询的分析和理解的研究已经成为广大研究者关心的问题。在这里，我们分析总结了查询分析的相关研究工作及研究成果。

4.1 查询优化

查询优化通过自动处理用户查询经常出现错误形式或表达不恰当，例如拼写错误、词形不当、缺少合适的操作符（如引号）、缩写不当等，来优化检索性能。

拼写错误是文本中常见的错误，而它也同样存在于用户查询之中。由于用户查询通常非常简短，缺乏足够的上下文信息，对纠正查询拼写错误的研究提出了新的难题。库色赞（S. Cucerzan）等人[19] 提出了利用用户查询日志来进行查询拼写修改。他们认为大量的用户查询日志本身为我们提供了查询这种语言隐式或显式的信息，利用查询日志上的语言统计信息作为指导，可以把用户查询一步一步修改为更合适的形式。他们在文章中提出了一种基于噪音-信道模型的迭代的查询修改方法，这种方法可以对查询中从简单到困难的各种错误依次进行修改。李（音译 M. Li）等人则利用了分布相似度来进一步改进查询拼写修改的方法[20]。

他们指出，正确的词和它可能的拼写错误之间的分布相似度是很高的，而不相关的两个词之间的分布式相似度则比较低。因此，分布相似度是判断一个词是否是另一个词的正确写法的有效依据，而用户查询日志恰恰是用来估计这种分布相似度的极好的资源。引文[21]中，作者把查询优化作为为结构化预测问题建模，提出了一个一体化区分式查询优化模型 CRF-QR（Conditional Random Fields for Query Refinement，用于查询优化的条件随机场模型），挖掘和利用查询日志来校正查询中的各种错误形式，达到优化查询的目的，克服了以往优化方法只关注一种优化任务的缺点。

4.2 查询冗余分析

用户查询通常只包含 2-3 个关键词，但趋势表明用户查询的平均长度在逐渐增加。长查询通常包含更加丰富的上下文信息，更加详细地描述了用户的查询需求，但是同时也包含一些噪音（例如，冗余（无关）的信息），导致面向长查询的搜索相关性更差，不能很好地满足用户需求。如何针对长查询自动构建更加简洁的查询成为当前各大搜索引擎关注的焦点之一。 查询冗余（无关）词删除技术通过删除这些冗余（无关）信息来获取更加简洁的查询以提高检索的性能。

引文[22]提出采用查询词之间的互信息来给所有的子查询排序。具体来说，首先根据查询词之间的互信息（基于语料集）构建一个全连通图，然后采用最大生成树算法来获取每个子查询的信息熵进行排序。引文[23]的基本思想是，针对不同的用户查询相应地选择查询扩张或删除可以极大地提高查询性能。作者基于引文[22]的用户交互思想，提出结合查询扩张和查询删除的子查询以备用户选择。

4.3 查询解析

用户查询包含了很多概念与知识，对用户查询进行解析，识别出查询中的关键概念（Key Concept）、命名实体（Named Entity）等深层信息，可以帮助我们更好地分析理解用户查询，有效地提高信息检索的智能化程度。

引文[24]基于查询相关（query-dependent）、语料相关（corpus-dependent）和语料无关（corpus-independent）等特征，采用机器学习的方法来识别长查询中的关键概念。学习算法采用了 AdaBoost.M1 元分类器，以 C4.5 为基本分类器。文章最后通过一个概率模型将所识别的概念结合到排序模型中来提高检索的性能。引文[25]研究了查询中的命名实体识别，通过对用户查询包含的命名实体的检测与分类来进一步解析用户查询中的语义单元以辅助检索。通过把查询中命名实体识别建模为最优三元组的求解问题，作者提出了一个新颖的概率模型，结合对大规模查询日志的挖掘和学习来有效地完成查询中的命名实体识别。

4.4 查询扩展

传统的查询扩展方法主要有基于相关词词表的全局技术、基于相关反馈或者伪相关反馈的局部技术以及全局与局部相结合的技术。崔（音译， H. Cui）等人在文献[26]中研究了如何利用用户查询日志来进行查询扩展。这种方法的主要思想是基于用户查询日志中的会话信息，对用户查询词和文档词语之间相关概率关系建模，并利用这种概率关系来为新的查询选取高质量的扩展词语。在文献[27]中，作者进一步探索了这种基于用户查询日志进行查询扩展的方法对于不同类型查询（长查询和短查询）的扩展效果。实验结果表明，利用用户查询日志进行的查询扩展方法能够显著地超过传统的基于词表和伪相关反馈的方法。

4.5 查询分类

对用户查询的分类包括了对查询意图的分类以及对查询话题的分类。对于查询分类的研究经历了逐步由浅入深的发展过程——从基于人工制定的规则对查询分类到利用查询日志和机器学习的方法来对查询进行分类。

布罗德（A. Broder）在 2002 年把互联网信息检索中用户意图归纳为三类[28]：导航类、信息类和事务类。对用户查询意图的这种分类方法具有重要的意义，但是这种分类的粒度相对较大。如何利用机器自动对用户查询分类是广大研究者关心的问题，沈（音译， Dou Shen）等人提出了利用中间分类体系作为对查询自动分类的概率框架[29]，这样的一个概率框架主要是以中间分类体系作为桥梁，来将用户查询映射到目标类别中去。

4.6 查询推荐

为了更好地辅助用户表达其查询意图，进一步提高用户的查询体验，查询推荐已经成为搜索引擎广泛采用的一项重要技术。查询推荐的研究工作长期以来受到大家的关注，较早的工作主要是基于文本内容，如今查询推荐更多地依赖于用户查询日志的用户数据信息。

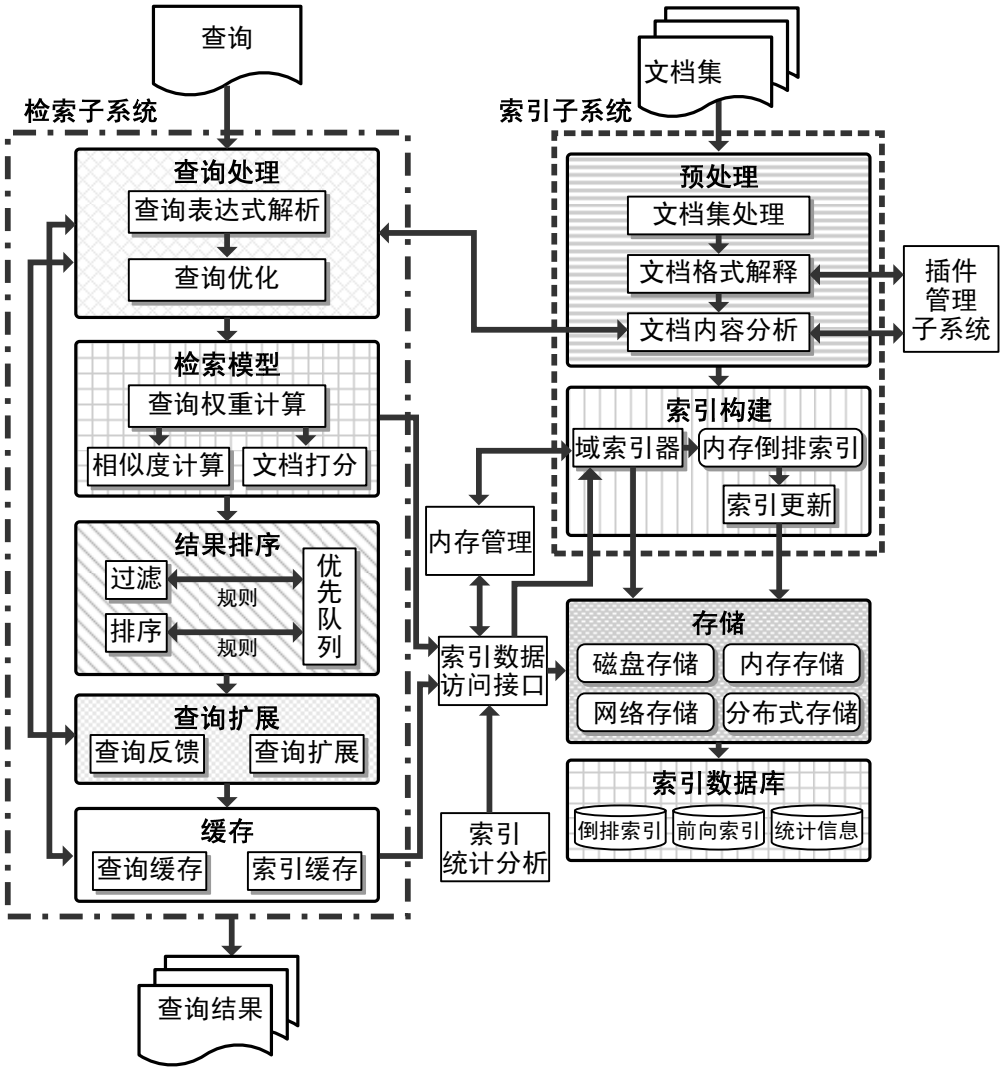

图3. FirteX 系统结构

在文献[30]中，冯赛卡（B. M. Fonseca）等人通过挖掘用户查询日志中的关联规则来进行查询推荐。张志勇（音译，Z. Zhang）等人挖掘了用户查询日志中的查询序列信息来进行查询推荐[31]。一方面，他们研究了用户检索过程中的序列行为并对其建模，用于描述同一

个会话中查询之间的相似关系；另一方面，他们还为“查询对”计算了基于内容的相似度。这个相似度使用了余弦距离的形式。通过把这两个相似度进行线性叠加，来对推荐的查询进行排序。此外，黄（音译，Chien-Kang Huang）等人在利用查询日志进行查询推荐时考虑了查询会话中上下文的信息[32]，他们认为推荐的查询应该是和整个查询会话相关的，而不仅仅是和最近的一个查询相关。

5 Firt eX 系统设计

FirteX 是我们开发的一套支持大规模文本的在线索引和检索平台。FirteX 区别于其他检索实验系统的主要特点是提供了一个动态文本在线索引和检索框架。利用它，研究者可以很方便地实现增量文本、动态文本的在线索引和检索，目前已经实现了包括动态平衡树策略在内的若干索引更新策略。此外，FirteX 不仅支持检索模型、查询反馈、自然语言处理等，而且也可以方便地实现索引更新算法、查询处理算法、分布式信息检索、查询缓存、内存管理，能够支持 TB 规模数据的检索要求。图 3 给出了 FirteX 的系统结构。

6 结束语

随着信息技术的发展，人们需要处理和检索的数据规模越来越大，需要更快地响应数据的变化。如何查询得更多、更快、更准，以及如何提供更具智能化的检索技术，成为信息检索面临的挑战。有鉴于此，本文介绍了信息检索中使用的索引组织、检索模型、查询分析等关键技术。针对目前信息检索技术的发展，本课题组开发了高效开源检索系统 FirteX，该系统提供了对在线索引等的支持，下一步我们将对系统支持的数据规模、检索模型等方面进行进一步改进。

参考文献：

[1] Baeza-Yates, Ribiero-Neto. 1999. Modern Information Retrieval. New York: ACM Press.

[2] Witten, I. H., Moffat, A. & Bell, T. C., Managing Gigabytes: Compressing and Indexing Documents and Images, second edn, Morgan Kaufmann, San Francisco, California, 1999.

[3] E. Fox and W. Lee. FAST-INV: A fast algorithm for building large inverted les. Technical Report TR 91-10, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1991.

[4] S. Heinz and J. Zobel. Efficient single-pass index construction for text database. Jour. Of the American Society for Information Science and Technology, 54(8):731-729, 2003.

[5] A. Moffat and T. A. Bell. In situ generation of compressed inverted files. Jour. of the American Society for Information Science, 46(7):537-550, 1995.

[6] Nicholas Lester. Efficient Index Maintenance for Text Database. PhD thesis, Department of Computer Science and Information Technology, RMIT, 2006.

[7] N. Lester, J. Zobel, and H.E. Williams, Efficient online index maintenance for text retrieval systems, Information Processing & Management, 42(4):916-933, July 2006.

[8] S. Büttcher and C. L. A. Clarke. Indexing time vs. query time trade-offs in dynamic information retrieval systems. In Proceedings of the 14th ACM Conference on Information and Knowledge Management (CIKM 2005). Bremen, Germany, November 2005.

[9] N. Lester, A. Moffat, and J. Zobel. Fast on-line index construction by geometric partitioning. In Proceedings of the ACM CIKM Conference on Information and Knowledge Management, pages 776-783, November 2005.

[10] Ruijie Guo, Xueqi Cheng, Hongbo Xu, Bin Wang. “Efficient On-line Index Maintenance for Dynamic Text Collections by Using Dynamic Balancing Tree”. In Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007). Lisbon, Portugal, November, 2007.

[11] S. Robertson. Overview of the okapi projects. In Journal of Documentation, pages 275–281, 1998.

[12] Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Stanford InfoLab (1999)

[13] Kleinberg, Jon (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM 46 (5): 604–632.

[14] K. Crammer and Y. Singer. Pranking with ranking. Advances on Neural Information Processing System, Pages 641-647, MIT Press, 2001.

[15] P. Li, C. Burges, et al. McRank: Learning to Rank Using Classification and Gradient Boosting, NIPS 2007.

[16] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to Rank Using Gradient Descent. In ICML, pages 89-96, 2005.

[17] Y. Freund, R. Iyer, R.E. Schapire and Y. Singer. An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research, 4:933–969, 2003.

[18] R. Herbrich, Thore Graepel, and Klaus Obermayer, Large Margin Rank Boundaries for Ordinal Regression, MIT Press, Cambridge, MA, 2000.

[19] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In Proceedings of EMNLP 2004, pages:293-300, 2004.

[20] M. Li, M. Zhu, Y. Zhang, and M. Zhou. Exploring distributional similarity based models for query spelling correction. In Proceedings of COLING-ACL 2006, pages:1025-1032, 2006.

[21] J. Guo, G. Xu, H. Li, and X. Cheng. A Unified and Discriminative Model for Query Refinement. In SIGIR ’08, pages 379–386, New York, NY, USA, 2008. ACM.

[22] G. Kumaran and J. Allan. A case for shorter queries, and helping users create them. In HLT-EMNLP Conference Proceedings, pages 220–227, Rochester, NY, 2007.

[23] Kumaran, G. and Allan, J. 2008. Effective and efficient user interaction for long queries. In SIGIR '08. ACM, New York, NY, pages:11-18.

[24] Bendersky, M. and Croft, W. B. 2008. Discovering key concepts in verbose queries. In SIGIR '08. ACM, New York, NY, pages:491-498.

[25] J. Guo, G. Xu, X. Cheng, and H. Li. Named Entity Recognition in Query. In SIGIR ’09, pages 267–274, Boston, MA, USA, 2009. ACM.

[26] H, Cui, J.R. Wen, J.Y. Nie and W.Y. Ma. Probabilistic Query Expansion Using Query Log. In Proceedings of the 11th international conference on World Wide Web 2002, Honolulu, Hawaii, USA, May 07-11, 2002.

[27] Hang Cui, Ji-Rong Wen, and Jian-Yun Nie. Query expansion by mining user logs. IEEE Trans. on Knowl. and Data Eng., 15(4):829-839, 2003. Member-Ma,, Wei-Ying.

[28] Broder, A. A Taxonomy of Web Search. SIGIR Forum 36(2), 2002.

[29] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Building Bridges for Web Query Classification. In SIGIR ’06, August 6-11, 2006, Seattle, Washington, USA.

[30] Bruno M. Fonseca, Paulo B. Golgher, Edleno S. de Moura, and Nivio Ziviani. Using association rules to discover search engines related queries. In LA-WEB '03, page 66, Washington, DC, USA, 2003. IEEE Computer Society.

[31] Zhiyong Zhang and Olfa Nasraoui. Mining Search Engine Query Logs for Query Recommendation. In WWW'06, Edinburgh, Scotland May 23 - 26, 2006

[32] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. J. Am. Soc. Inf. Sci. Technol., 54(7):638-649, 2003.

[33] 郭瑞杰.大规模动态文本在线索引技术研究(博士论文),2008.

作者简介：

俞晓明： 中国科学院计算技术研究所，助理研究员，Email: yuxiaoming@software.ict.ac.cn
郭嘉丰： 中国科学院计算技术研究所， 助理研究员
朱小飞： 中国科学院计算技术研究所， 博士生
关　峰： 中国科学院计算技术研究所， 助理研究员
程学旗： 中国科学院计算技术研究所， 研究员

（ 上接第 60 页 ）

[22] 李保利、愈士汶. 话题识别和跟踪研究. 计算机工程与应用, 2003.

[23] 骆卫华、于满泉、许洪波、王斌、程学旗. 基于多策略优化的分治多层聚类算法的话题发现研究. 中文信息学报, 2005.

[24] 骆卫华、刘群、程学旗. 话题检测与跟踪技术的发展与研究. 全国计算语言学联合学术会议, 2003.

[25] 洪宇、张宇、刘挺、李生. 话题检测与跟踪的评测及研究综述. 中文信息学报, 2007.

作者简介：

张　瑾： 中国科学院计算技术研究所 网络重点实验室 助理研究员
Email:zhangjin@software.ict.ac.cn
杨　森： 中国科学院计算技术研究所 网络重点实验室 硕士生
王孝宗： 中国科学院计算技术研究所 网络重点实验室 硕士生
罗　维： 中国科学院计算技术研究所 网络重点实验室 硕士生
杜　攀： 中国科学院计算技术研究所 网络重点实验室 博士生
程学旗： 中国科学院计算技术研究所 网络重点实验室 研究员